Dialogue et representation des informations dans un systeme de messageries intelligent

Philippe Besnard, René Quiniou, Patrice Quinton, Patrick Saint Dizier, Jacques Siroux, Laurent Trilling

INRIA

CENTRE DE RENNES

IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.: 954 90 20

Rapports de Recherche

N° 193

DIALOGUE ET REPRÉSENTATION DES INFORMATIONS DANS UN SYSTÈME DE MESSAGERIE INTELLIGENT

Philippe BESNARD
René QUINIOU
Patrice QUINTON
Patrick SAINT-DIZIER
Jacques SIROUX
Laurent TRILLING

Février 1983

Dialogue et représentation des informations dans un système de messagerie intelligent

Philippe BESNARD
René QUINIOU
Patrice QUINTON
Patrick SAINT-DIZIER
Jacques SIROUX
Laurent TRILLING

Abstract

Data representation and data evolution remains a crucial problem in Artificial Intelligence. Solutions using production systems, relational databases and their logical formalisation and non-monotonic logic (especially default logic) are given and used to formalize an application from the office automation domain (CIGARE a system designed to help the schedule of meetings).

Résumé

La représentation des données et leur évolution demeure un problème crucial en Intelligence Artificielle. Des solutions utilisant les systèmes de production, les bases de données relationnelles et leur formalisation en logique, ainsi que la logique non-monotone (la logique des défauts en particulier) sont proposées et utilisées pour formaliser une application bureautique (CIGARE un système de gestion automatisée de rencontres).

INTRODUCTION

La représentation des informations et leur évolution demeure un problème crucial en intelligence artificielle. Au travers d'une application bureautique nous avons essayé d'apporter des solutions empruntées aux systèmes de production, aux bases de données relationnelles formalisées en logique et à la logique non-monotone.

Le but du système CIGARE est de procurer une aide dans l'organisation de réunions. Le rôle du système est de prendre en charge les contacts nécessaires à l'organisation d'une réunion, de recueillir les contraintes temporelles des personnes concernées, d'élaborer une solution satisfaisant le plus grand nombre de personnes et enfin de négocier avec les personnes ne pouvant participer s'il y a lieu.

Les caractéristiques essentielles de cette application sont les suivantes:

-informations incertaines: les informations (manipulées par le système) sont de nature évolutive. Le temps nécessaire à l'organisation d'une réunion étant important (de quelques heures à plusieurs jours) certains utilisateurs peuvent être amenés à modifier leur avis en fonction de contraintes propres.

-acceptabilité: elle est liée à la souplesse des procédures du dialogue permettant d'accéder au système. En outre, une règle essentielle est que le système dérange le moins possible les utilisateurs. Il faut donc que le système prenne en compte des spécifications de comportement données par les utilisateurs. Par ailleurs, les situations d'exception à ces règles doivent être facilement traitées.

Ce qui suit est une tentative de synthèse de ces différents aspects qui concernent directement le système CIGARE.
Le rapport est organisé de la façon suivante.

Nous présentons au chapitre I, sur des exemples tirés de l'application CIGARE, le modèle relationnel pour les bases de données.

Après un rappel sur la logique du 1er ordre au chapitre II, nous présentons au chapitre III un système d'interrogation de base de données utilisant un démonstrateur de théorèmes. Nous présentons également les différentes représentations d'une base de données relationnelle en logique.

Après une présentation, au chapitre IV, des principes de la logique non-monotone et des outils permettant de la traiter, nous nous attardons particulièrement sur la logique des défauts. Cette logique permet de prendre en compte efficacement l'évolution d'une théorie représentant une base de données lors de la mise à jour de celle-ci.

Le chapitre V concerne plus directement CIGARE, vu comme un système expert. Il comporte comme la plupart des systèmes de production une base de connaissances (ensemble de règles de production), un interpréteur et une base de données (base de faits). La base de données est représentée par une base de données relationnelle.

Enfin nous présentons au chapitre VI l'interface d'accès en langage naturel du système CIGARE.

I RAPPEL SUR LES BASES DE DONNEES RELATIONNELLES

Par définition, une base de données relationnelle doit satisfaire le modèle relationnel (CODD [1970]). Ce dernier se définit comme suit.

Un domaine est un ensemble de valeurs du même type.
Soient $D_1, D_2, \ldots, D_n$, n domaines (pas nécessairement distincts).
Une relation définie sur $D_1, \ldots, D_n$ est un sous-ensemble du produit cartésien $D_1 x \ldots x D_n$. Une relation est donc un ensemble de n-uplets $(d_1, \ldots, d_n)$ tels que tout d_i soit élément du domaine D_i.
L'index de chaque composant d'un n-uplet est appelé attribut.
Une base de données relationnelle est une collection évolutive de relations.

Exemple:

$D_1 = \{108, 110, 112, 220, 224\}$
$D_2 = \{Anne, Bernard, Claude\}$
$D_3 = \{lundi, mardi, mercredi, jeudi, vendredi\}$
$D_4 = \{administration, enseignement, recherche\}$

REUNIONS

INVITE	JOUR	SALLE
Anne	mercredi	220
Claude	mercredi	220
Bernard	vendredi	112
Claude	vendredi	112

Cette mini-base de données relationnelle ne contient qu'une relation, REUNIONS, qui possède 3 attributs, INVITE, JOUR et SALLE. Les 4 triplets de la relation REUNIONS expriment qu'une réunion se tiendra mercredi en salle 220 avec Anne et Claude, et qu'une autre réunion est prévue pour vendredi, en salle 112 avec Bernard et Claude.

Il n'existe pas de liens structurels tels que des pointeurs entre les relations. Les associations entre relations sont représentées uniquement par des valeurs et sont exploitées par des opérateurs de haut niveau.

L'algebre relationnelle fait aussi partie du modele relationnel. Elle sert a définir les opérateurs de manipulation des relations. Les relations étant des ensembles, les opérateurs classiques d'union, d'intersection et de différence (d'ensembles) sont applicables (dans les limites de compatibilité des relations auxquelles l'opérateur s'applique: les attributs de chaque

relation doivent correspondre un à un). Il existe d'autres opérations qui sont la restriction, la projection, la theta-jointure, la jointure et la division.

I 1 La restriction

La restriction (opération unaire) ne conserve que les n-uplets de la relation qui satisfont la propriété désirée.

Exemple:

REUNIONS[INVITE#Claude]

INVITE	JOUR	SALLE
Anne	mercredi	220
Bernard	vendredi	112

restriction de la relation REUNIONS pour la propriété INVITE#Claude

I 2 La projection

La projection (opération unaire) élimine les attributs non spécifiés.

Exemple:

REUNIONS[INVITE,JOUR]

INVITE	JOUR
Anne	mercredi
Claude	mercredi
Bernard	vendredi
Claude	vendredi

projection de REUNIONS sur les attributs INVITE et JOUR

I 3 La jointure et la theta-jointure

La theta-jointure (opération binaire) concatène les n-uplets de 2 relations s'ils vérifient l'un vis-à-vis de l'autre la propriété spécifiée. La jointure coincide avec le cas où cette propriété est l'égalité.

Exemple: Soient les relations
PROJET DE REUNIONS

INVITE	SALLE
Anne	105
Bernard	105
Claude	220
Anne	220

RESERVATION

SALLES	JOUR
200	lundi
100	mardi

La relation RESERVATION donne les numéros à partir desquels les salles ne sont plus réservées. Le premier doublet signifie que lundi les salles dont le numéro est inférieur à 200 sont réservées.
Soit la propriété SALLE>SALLES. La theta-jointure des relations PROJET DE REUNIONS et RESERVATION par cette propriété est alors la relation
PROJET DE REUNIONS[SALLE>SALLES]RESERVATION

INVITE	SALLE	JOUR	SALLES
Anne	105	mardi	100
Bernard	105	mardi	100
Claude	220	lundi	200
Anne	220	lundi	200
Claude	220	mardi	100
Anne	220	mardi	100

I 4 La division

La division (opération binaire) sélectionne les n-uplets d'une relation qui ont un composant (relatif à l'attribut spécifié) égal au composant (relatif à l'autre attribut spécifié) d'un n-uplet de la deuxième relation. La relation résultante est amputée de l'attribut sur lequel s'est faite la division.

Exemple: Soient les relations

REUNIONS

INVITE	JOUR	SALLE
Anne	mercredi	220
Claude	mercredi	220
Bernard	vendredi	112
Claude	vendredi	112

JOUR_POSSIBLE

JOUR
mercredi
jeudi

La division de la relations REUNIONS par la relation JOUR_POSSIBLE relativement aux attributs JOUR, donne la relation

REUNIONS[JOUR/JOUR]JOUR_POSSIBLE

INVITE	SALLE
Anne	220
Claude	220

Enfin, le modèle relationnel comporte des règles d'intégrité qui décrivent les contraintes que doivent respecter les informations de la base de données.

II RAPPEL SUR LA LOGIQUE DU PREMIER ORDRE

II 1 Le langage

Pour établir la syntaxe du langage de la logique du premier ordre, MENDELSON [1964] part du principe que ce langage s'appuie sur quelques symboles primitifs:

- -Les constantes
- -Les variables
- -Les symboles fonctionnels
- -Les symboles relationnels (ou predicats)
- -Les connecteurs logiques ~ (unaire) et => (binaire)
- -Le quantificateur universel

A partir de ces symboles, vont être définis

-Les termes

(1) Toute constante est un terme
(2) Toute variable est un terme
(3) $f(t_1,\ldots,t_n)$ est un terme si $t_1,\ldots,t_n$ sont des termes et si f est un symbole fonctionnel n-aire
(4) Les cas (1) à (3) décrivent les seuls termes existants

-Les formules atomiques

$P(t_1,\ldots,t_n)$ est une formule atomique si P est un symbole relationnel n-aire et si $t_1,\ldots,t_n$ sont des termes

Les formules bien-formées de la logique du premier ordre sont

(1) Les formules atomiques
(2) Les formules (x) P où P est une formule bien-formée et x une variable
(3) Les formules ~P et P => Q si P et Q sont des formules bien-formées
(4) Les cas (1) à (3) décrivent les seules formules bien-formées existantes

Dans la suite, le mot formule référera à l'expression formule bien-formée.

(Ex) P est une notation commode pour ~((x) ~P).

Une formule dont toutes les variables sont quantifiées est close.

II 2 L'approche sémantique: la théorie des modèles

Contrairement à LYNDON [1964], mais similairement à la plupart des logiciens, MENDELSON [1964] restreint l'évaluation de valeurs de vérité aux formules closes.

Ainsi une interprétation est la donnée

- -d'un domaine E non vide
- -d'une application qui à chaque constante associe un élément de E

-pour chaque symbole fonctionnel d'arité n, d'une application de $E^n \to E$
-pour chaque symbole relationnel d'arité n, d'une application de $E^n \to \{vrai, faux\}$

Dans une interprétation, une formule close est donc soit vraie (ou satisfaite), soit fausse.

Une interprétation d'un ensemble de formules W est un modèle de W ssi chaque formule de W est vraie dans cette interprétation.

Une formule est valide si elle est vraie dans toute interprétation.

Une formule w est conséquence sémantique d'un ensemble de formules W ssi w est vraie dans tout modèle de W.
Notation: $W \models w$

II 3 L'approche syntaxique: la théorie de la démonstration

Une règle d'inférence permet, sur des considérations syntaxiques (c'est-à-dire par manipulation de symboles), de déduire une formule à partir d'un ensemble de formules.

Règles d'inférence

-Modus ponens
pour toutes formules P et Q
à partir de P et P => Q inférer Q
Notation: $P, P \Rightarrow Q \vdash Q$

-Généralisation
pour toute formule P, à partir de P inférer (x) P
Notation: $P \vdash (x)\ P$

Un schéma d'axiomes permet de dériver des formules constituées conformément à ce schéma.

Exemple de schéma d'axiomes:
pour toutes formules P et Q
la formule $P \Rightarrow (Q \Rightarrow P)$ est un axiome

Les 5 schémas d'axiomes appelés axiomes logiques et les 2 règles d'inférence ci-dessus constituent la théorie de la démonstration pour le calcul des prédicats du premier ordre.

II 4 Consistance Complétude Décidabilité

La consistance stipule que de 2 formules contradictoires (w et $\tilde{}w$), l'une au plus peut être démontrée. Cette propriété du calcul des prédicats du premier ordre, démontrée par HILBERT et ACKERMANN [1928], se note

(1) $W \vdash w \Rightarrow W \models w$ w étant une formule et W un ensemble de formules

La complétude établit que pour toute formule w qui est conséquence sémantique d'un ensemble de formules W, il existe une démonstration de w à partir des hypothèses W. Démontrée par GODEL [1930], cette propriété se note

(2) $W \models w \Rightarrow W \vdash w$

Les résultats de consistance et de complétude établissent l'équivalence de la théorie des modèles et de la théorie de la démonstration. Les approches sémantiques et syntaxiques représentent donc 2 méthodes équivalentes de raisonnement.

(1) et (2) $W \vdash w \Leftrightarrow W \models w$

Soit une procédure à laquelle on soumet une formule quelconque en entrée et qui, au bout d'un temps fini, indique en réponse s'il existe ou non une démonstration (sans la fournir nécessairement s'il y en a une) de cette formule. Le problème de l'existence d'une telle procédure s'appelle le problème de décision. Pour le calcul des prédicats du premier ordre, le problème de décision est insoluble, autrement dit, le calcul des prédicats du premier ordre est indécidable. En effet, CHURCH [1936] et TURING [1936] ont montré qu'il n'y a pas de procédure capable de déterminer en un temps fini, si une formule est valide ou non. Cette indécidabilité interdit la mise en oeuvre de cette théorie, puisque corollairement elle démontre l'existence de calculs infinis (Pour tester la validité d'une formule, la soumettre à l'ordinateur peut conduire celui-ci à ne jamais s'arrêter). Heureusement, à l'intérieur du calcul des prédicats, il est possible d'isoler certaines théories axiomatiques qui admettent une solution au problème de décision.

II 5 Les théories axiomatiques

Une théorie formelle axiomatique satisfait les 4 conditions ci-dessous:

- -Les expressions considérées appartiennent à un langage donné (ici celui du calcul des prédicats du premier ordre)
- -Un sous-ensemble de ces expressions constitue l'ensemble des formules bien-formées
- -L'ensemble des axiomes est un sous-ensemble des formules bien-formées
- -Les seules relations entre formules sont des règles d'inférences

Les théorèmes sont les formules déduites des axiomes grâce aux règles d'inférence.

C'est la deuxième condition qui mène à des théories décidables. Il suffit pour cela de restreindre suffisamment l'ensemble des formules bien-formées. Par exemple, le calcul des prédicats monadiques (où les symboles relationnels sont d'arité inférieure ou égale à 1) est décidable (LOWENHEIM [1915]).

Dans la suite, seules seront considérées les théories consistantes et complètes, les autres offrant peu d'intérêt.

Dans le cadre des théories du premier ordre, plusieurs types de problèmes de décision se posent. Soient entre autres questions

1-Une expression formelle est-elle une formule?

2-Une suite finie donnée de formules est-elle une démonstration?

3-Une formule donnée est-elle démontrable?

Pour les questions de type 1, la décomposition de l'expression en prédicats et en termes apporte une réponse.

Pour les questions de type 2, la procédure consiste à considérer chaque formule en commencant par la première et à vérifier qu'elle découle des précédentes par l'application d'une règle d'inférence.

Le problème de l'existence d'une procédure permettant de répondre aux questions de type 3 est le problème de décision tel qu'il a été exposé dans le paragraphe précédent. Cependant, bien que conformément à l'indécidabilité du calcul des prédicats du premier ordre ce problème soit insoluble pour l'ensemble des théories du premier ordre, il existe un certain nombre de théories décidables (Une théorie décidable est une théorie pour laquelle existe une procédure permettant de répondre à toute question de type 3 en un temps fini. Noter qu'une théorie peut avoir été prouvée décidable sans qu'une telle procédure ait été exhibée. La démonstration de la décidabilité d'une théorie constitue l'aspect mathématique de la solution du problème de décision pour cette théorie et la découverte d'une procédure adéquate en est l'aspect informatique).

YAZDANIAN [1976] effectue un recensement des classes de formules du premier ordre dont soit la décidabilité soit l'indécidabilité a été démontrée.

III UN SYSTEME D'INTERROGATION DE BASE DE DONNEES

III 1 Modèlisation conceptuelle

Définir une base de données passe par la définition d'un modèle conceptuel qui pose les bases de la représentation logique (par opposition à physique) de l'information. (Pour un éventail de positions sur la modélisation conceptuelle, voir BRODIE et ZILLES [1981]).
L'approche relationnelle (CODD [1970]) de la spécification d'un modèle conceptuel comporte de nombreux liens avec la logique mathématique. Bien que ce soit également vrai pour les simples systèmes de gestions de base de données, cela ouvre de prometteuses perspectives pour les systèmes de question-réponse qui opèrent sur des bases de données déductives (Une base de données est déductive lorsqu'elle intègre de l'information implicite, c'est-à-dire de l'information représentée de facon intensionnelle -intensionnelle signifiant ici désignée par une propriété-. Il arrive d'ailleurs qu'il soit question de base de données intensionnelle et de base de données extensionnelle). Pour s'en convaincre, il suffit de remarquer que

-La traduction du contenu d'une base de données relationnelle en formules du calcul des prédicats du premier ordre est directe

-Pour la déduction, la logique du premier ordre fournit une procédure de preuve saine et complète

Pour tout ce qui concerne les liens entre la logique et les bases de données relationnelles, consulter GALLAIRE et MINKER [1978].

III 2 Les modèles logiques

La distinction évoquée précédemment entre base de données intensionnelle et base de données extensionnelle est reflétée dans ce paragraphe par la séparation de l'information en lois générales et faits élémentaires.
L'expression loi générale désigne une information qui se traduit par plus d'un n-uplet.
L'expression fait élémentaire désigne une information qui se traduit par un seul n-uplet.

III 2 1 Approche par la théorie de la démonstration

Le premier modèle logique pour une base de données relationnelle consiste en une vue de la base de données comme d'une théorie axiomatique.
Les axiomes de cette théorie étant constitués des faits élémentaires et des lois générales de la base de données.

Exemple:

INVITE

NOM
Luc

CHERCHEUR

NOM
Anne
Claude
Denis

PRESENCE

NOM	DATE
Anne	lundi
Anne	jeudi
Claude	mardi
Denis	mardi
Luc	lundi

Faits élèmentaires

(x) CHERCHEUR(x) => INVITE(x)

Lois générales

Les lois générales jouent exclusivement le rôle de règles de déduction, c'est-à-dire qu'elles permettent de déduire de nouveaux faits élèmentaires. Les règles de déduction forment la base de données intensionnelle, alors que les faits élèmentaires constituent la base de données extensionnelle.

III 2 2 Approche par la théorie des modèles

Le deuxième modèle logique pour une base de données relationnelle consiste en une vue de la base comme d'une interprétation d'une théorie axiomatique.

Exemple:

INVITE

NOM
Anne
Claude
Denis
Luc

CHERCHEUR

NOM
Anne
Claude
Denis

PRESENCE

NOM	DATE
Anne	lundi
Anne	jeudi
Claude	mardi
Denis	mardi
Luc	lundi

Faits élémentaires

(x) CHERCHEUR(x) => INVITE(x)

Lois générales

Les lois générales font office de règles d'intégrité (cf. le modèle relationnel) et sont considérées comme les axiomes d'une théorie axiomatique.
Les faits élémentaires forment eux une interprétation de cette théorie.
En conséquence, l'utilisation des lois générales comme règles de déduction aboutirait à la production d'information redondante.

III 2 3 Une extension de l'approche par la théorie des modèles

En réalité, ce troisième modèle incorpore également l'aspect théorie de la démonstration. Le point de départ de l'élaboration de ce type de modèle repose sur la constatation que les règles de déduction se différencient des règles d'intégrité uniquement par l'interprétation qui en est prise. Ainsi les lois générales sont considérées dans leur totalité, comme règles d'intégrité dans le modèle précédent et comme règles de déduction dans le premier modèle. Les exemples choisis illustrent d'ailleurs parfaitement cette bivalence des lois générales.

Exemple:

INVITE

NOM
Luc

REUNION

NOM	DUREE
reunion comite projets	2 h

CHERCHEUR

NOM
Anne
Claude
Denis

PRESENCE

NOM	DATE
Anne	lundi
Anne	jeudi
Claude	mardi
Denis	mardi
Luc	lundi

Faits élémentaires

(x) CHERCHEUR(x) => INVITE(x)

Règles de déduction

(x) (y) REUNION(x,y) => y>15mn

Règles d'intégrité

Dans cette troisième approche, les lois générales sont partagées en règles d'intégrité et règles de déduction. Les faits élémentaires constituent, avec les règles de déduction, une interprétation de la théorie axiomatique dont les règles d'intégrité sont les axiomes.
Ce modèle, tout comme le premier, présente l'avantage de pouvoir représenter les relations en partie explicitement et en partie implicitement (La relation INVITE dans l'exemple).

III 2 4 Spécificités des 3 modèles logiques

L'approche par la théorie de la démonstration, où la base de données est vue comme une théorie, est plus riche parce que la base satisfait alors un ensemble de modèles, tandis que dans les 2 autres approches la base n'est elle-même qu'une interprétation. Concrètement, cet avantage se traduit, pour les bases de données vues comme des théories, par leur aptitude à enregistrer des informations disjonctives (définies et discutées plus loin).

La logique du premier ordre fournit généralement le langage d'interrogation des bases de données modélisées logiquement. En

revanche, les 2 familles (théorie/interprétation) de ces modèles divergent au niveau de l'évaluation des questions.
Si la base de données est vue comme une théorie, les questions sont des formules interprétées comme des théorèmes à prouver (Ce sont les questions de type 3).
Autrement, les questions doivent être évaluées directement sur l'interprétation (qui est constituée par la base de données elle-même). Cette opération s'apparente à la méthode des tables de vérité.

III 3 Un démonstrateur de théorèmes

Les modèles logiques faisant intervenir les théories font appel à des procédures pouvant déterminer si une formule est ou non un théorème. Ces procédures sont qualifiées de procédures de preuve et leur mise en oeuvre de démonstrateurs de théorèmes.

III 3 1 Réfutation

Un ensemble de formules est inconsistant s'il n'a aucun modèle.

Un littéral est une formule atomique éventuellement niée.

Une clause est une disjonction de littéraux dont toutes les variables sont quantifiées universellement.

Toute formule du calcul des prédicats du premier ordre peut se mettre sous forme d'un ensemble de clauses (Pour une description du procédé, voir CHANG et LEE [1973]).

Les 2 théorèmes suivants permettent de relier la démonstration d'une formule à la vérification de l'inconsistance d'un ensemble de clauses.

Théorème: Etant donné une formule w et un ensemble de formules W, w est conséquence sémantique de W ssi l'ensemble de formules W U {~w} est inconsistant

Théorème: Etant donné un ensemble de formules W et S l'ensemble des clauses correspondantes, W est inconsistant ssi S est inconsistant

Une réfutation est une démonstration de l'inconsistance d'un ensemble de clauses.

Pour prouver qu'une formule w est un théorème d'une théorie W, il suffit donc de trouver une réfutation de l'ensemble des clauses correspondant à W U {~w}

Un troisième théorème fournit un critère simple pour tester l'inconsistance d'un ensemble de clauses:

Théorème: Un ensemble de clauses est inconsistant si il contient la clause vide

III 3 2 Résolution

La résolution est une règle d'inférence travaillant sur des clauses et dûe à ROBINSON [1965].

Une substitution est un ensemble d'applications qui à une variable associent un terme.

Exemple: soit $\Theta=\{(t_1,x_1),\ldots,(t_m,x_m)\}$
L'application de la substitution Θ au littéral $P(x_1,\ldots,x_n)$ donne le littéral $P(t_1,\ldots,t_m)$

2 littéraux L_1 et L_2 sont unifiables s'il existe une substitution Θ telle que $L_{1\Theta}=L_{2\Theta}$

Exemple: soient P(f(x),y) et P(z,b) où x,y,z sont les variables
Ces 2 littéraux sont unifiables par $\Theta=\{(f(x),z),(b,y)\}$

La substitution λ est l'unificateur le plus général de 2 littéraux L_1 et L_2 si pour tout unificateur Θ de L_1 et L_2 il existe une substitution γ telle que

$$(L_{1\lambda})_\gamma = L_{2\Theta}$$

L'unificateur le plus général peut se voir comme l'unificateur qui instancie le moins de variables possible.

Exemple: $\Theta=\{(g(a),z),(b,y)\}$ est un unificateur de
P(g(x),y) et P(z,b)
Mais l'unificateur le plus général est
$\{(g(x),z),(b,y)\}$

Soient 2 clauses $C=L_1 v\ldots vL_i v\ldots L_n$ et $C'=L'_1 v\ldots vL'_j v\ldots vL'_m$
La résolution peut s'appliquer à C et C' si L_i et $\neg L'_j$ sont unifiables, c'est-à-dire s'il existe un unificateur le plus général λ de L_i et $\neg L'_j$. La résolvante de C et C' est alors la clause

$$L_{1\lambda} v\ldots vL_{i-1\lambda} vL_{i+1\lambda} v\ldots vL_{n\lambda} vL'_{1\lambda} v\ldots vL'_{j-1\lambda} vL'_{j+1\lambda} v\ldots vL'_{m\lambda}$$

Exemple: soient P(g(x),y) v Q(x,y) et R(c,z) v ¬P(z,b)
$\{(g(x),z),(b,y)\}$ est l'unificateur le plus général
La résolvante est Q(x,b) v R(c,g(x))

Une règle d'inférence est saine ssi toute formule inférée au moyen de cette règle est conséquence sémantique des hypothèses.

Une règle d'inférence est complète ssi toute formule w qui est conséquence sémantique d'un ensemble de formules W peut être inférée par l'application (éventuellement répétée) de cette règle sur W (éventuellement augmenté des formules inférées entre temps).

ROBINSON [1965] a démontré que la résolution était saine et complète.

III 3 3 Une classe de formules décidable pour la résolution

Une clause est à variable prédéfinies si

(1) Ses symboles fonctionnels sont des constantes

(2) Toute variable qui apparait dans un littéral positif apparait également dans un littéral négatif

Exemple: Sont des clauses à variables prédéfinies
P(x) v ~Q(x,y) et P(x,y,b) v ~Q(x) v ~R(y)
Ne sont pas des clauses à variables prédéfinies
P(x) et P(x) v Q(y)

BOSSU et SIEGEL [1981] ont montré que l'ensemble des clauses à variables prédéfinies était décidable pour la résolution.

III 3 4 Le saturateur

La méthode de résolution de BOSSU et SIEGEL [1981], appelée saturation, est fondée sur le A-ordering, stratégie de résolution développée par KOWALSKI et HAYES [1969].

Il existe donc un ordre sur les formules atomiques, qui est défini par
-un ordre sur les constantes (a<b<c...)
-un ordre sur les symboles relationnels (P<Q<R...)
-le signe du littéral n'intervient pas dans l'ordre sur les formules atomiques

Exemple: Si a<b (ordre sur les constantes)
et P<Q<R (ordre sur les prédicats), alors
R(a,a) < R(a,b)
P(b) < ~Q(a)
R(a,b) et R(x,a) ne sont pas ordonnables car a (une constante) et x (une variable) ne le sont pas

Cet ordre sert à définir le type des clauses (les clauses strictement ordonnées) sur lesquelles s'effectue la résolution:

(1) La clause vide

(2) Les clauses à variables prédéfinies, sans littéraux égaux, dont le littéral de tête est
-soit un littéral négatif
-soit un littéral positif inférieur à tous les autres littéraux de la clause et tel que toute variable qui apparait dans un autre littéral apparait également dans ce littéral de tête

Exemple: P(x,y) v ~Q(x) v ~R(y) est strictement ordonnée
P(x) v ~Q(x) v ~R(y) ne l'est pas car y n'apparait pas dans le littéral positif de tête

Le A-ordering limite la résolution au littéral de tête des clauses ordonnées (ici appelées clauses strictement ordonnées). Dans la résolution sur le littéral de tête, seul le littéral de tête de chaque clause est pris en compte pour l'unification. Si le choix de l'ordonnancement a été judicieux, les possibilités de produire des résolvantes sont réduites, ce qui explique l'intérêt du A-ordering.

Un ensemble C de clauses est saturé pour la résolution sur le littéral de tête, si toute résolvante sur le littéral de tête de 2 clauses de C, qui n'est pas une tautologie, est subsumée par une clause de C.

La saturation va consister à saturer un ensemble de clauses strictement ordonnées et à vérifier si cet ensemble saturé contient la clause vide ou non, ce qui permet de décider de son inconsistance ou de sa consistance.

BOSSU et SIEGEL [1981] ont montré qu'à partir d'un ensemble initial de clauses strictement ordonnées, seule une suite finie de clauses strictement ordonnées différentes peut être générée, avant d'aboutir à un ensemble saturé.
C'est ce qui confère le caractère de décidabilité à la saturation.

La saturation a mené à la réalisation d'un démonstrateur de théorèmes décidable, le saturateur.

III 4 Information incomplète

L'expression informations incomplètes recouvre la plupart des phénomènes dont il est difficile de rendre compte dans une base de données relationnelle. En gros, une information incomplète est une information qui ne peut se traduire par une conjonction de n-uplets.

III 4 1 Les informations disjonctives

Une information disjonctive se caractérise par la donnée d'un certain nombre de faits dont l'un est vrai (Lequel de ces faits est vrai restant inconnu). Par exemple, la phrase "La réunion aura lieu lundi ou mardi" ne permet pas de savoir si "La réunion aura lieu lundi" est vraie ou si "La réunion aura lieu mardi" est vraie, bien que l'une de ces 2 propositions soit vraie.

Le problème est lié à la récupération des faits élémentaires composant l'information disjonctive. En effet, ceux-ci ne sont ni vrais ni faux, c'est pourquoi une logique à 2 valeurs les traitent isolément incorrectement: Ils sont assimilés à des informations négatives.
Alors que la solution classique associe un modèle à la vérité de chacun de ces faits élémentaires (dans l'exemple, une interprétation où la réunion a lieu lundi et une autre où elle a

lieu mardi), LIPSKI [1979] a abordé ce problème au travers de la logique modale.

III 4 2 Les valeurs nulles

Dans un n-uplet, une valeur nulle est une valeur extérieure au domaine associé à l'attribut. Cette valeur peut s'interpréter de 2 facons.

Exemple: PRESENCE

NOM	DATE
Jean	....

L'unique doublet de cette relation exprime qu'il existe un jour (non précisé) où Jean sera présent

Le premier type de valeurs nulles concerne l'existence d'une entité inconnue qui satisfait une relation. Il s'agit alors d'un cas particulier des informations disjonctives. La seule différence est qu'ici l'attribut inconnu peut prendre toutes les valeurs du domaine et non plus seulement un sous-ensemble. Dans l'exemple précédent, si le domaine de l'attribut DATE est {lundi,...,vendredi}, l'information enregistrée au moyen de la valeur nulle est équivalente à la disjonction des 5 faits élémentaires exprimant que Jean sera présent lundi, mardi...

CODD [1979] a proposé le traitement qui parait le plus naturel. Un nouvel individu apparait, désignant une entité existante, bien qu'inconnue. Cependant, l'introduction de cette nouvelle entité conduit à ajouter une troisième valeur de vérité, dont la sémantique est approximativement "peut-être".

Les tables de vérité pour cette logique sont

val(vrai)=1
val(faux)=Ø
val(indéfini)=1/2
val(non(vrai))=Ø
val(non(faux))=1
val(non(indéfini))=1/2
val(P & Q)=inf(val(P),val(Q))
val(P v Q)=sup(val(P),val(Q))

Malheureusement, ceci entraine val(P v ~P)=1/2 pour P indéfini. Donc P v ~P est indéfini si P l'est. Les tautologies ne sont pas conservées. CODD [1979] justifie toutefois ce comportement en arguant que tout ce qui est obtenu à partir de constituants partiellement inconnus est lui-même partiellement inconnu.

Exemple:

REPONSE

NOM	POSITION	MOTIF REFUS
Edith	viendra	....
Yves	n'ira pas	congrès

Le premier triplet de cette relation exprime que Edith accepte d'assister à la réunion. Dans ce contexte, l'attribut MOTIF REFUS n'a plus de sens.

Le second type de valeurs nulles correspond au fait qu'il n'existe aucune valeur significative pour l'attribut adressé. La valeur nulle indique l'interdiction pour l'attribut de prendre une valeur dans le n-uplet concerné.

III 4 3 Information négative. Hypothèse de monde clos

La représentation des informations négatives s'est toujours révélée un problème crucial de la formalisation des bases de données relationnelles par la logique du premier ordre. D'autre part, l'appréhension classique des bases de données était de considérer que la base incluait toutes les informations pertinentes. Tout cela plaidant en faveur d'une représentation implicite des informations négatives, a conduit à l'hypothèse de monde clos (REITER [1978]), que 2 procédés permettent de réaliser.

III 4 3 1 La négation par échec

La négation par échec (CLARK [1978]) permet d'inférer ~P si P n'a pu être prouvé

$$\text{si } T \not\vdash P \text{ alors } T \vdash \sim P$$

Cette représentation des informations est la plus implicite possible puisque la négation par échec est une règle d'inférence. Malheureusement elle mène à des inconsistances en présence d'information disjonctive.

Exemple:

(x) ~Absent(x) => Present(x)
Absent(Pierre)
Invite(Jean)
Avec la négation par échec, la réponse à la question "Qui est présent?" (Ex Present(x)?) est "personne". Or la réponse à la question "Jean est-il présent?" est "oui".

Enfin la négation par échec, une formalisation de l'hypothèse de monde clos, traite incorrectement les valeurs nulles en les assimilant à des informations négatives.

III 4 3 2 Les règles de particularisation

Lorsque la base de données est vue comme une théorie, celle-ci regroupe trois sortes d'axiomes:

-les faits élémentaires
-les lois générales
-les règles de particularisation.

Les règles de particularisation forment l'approche axiomatique de la formalisation de monde clos. Les règles de particularisation comprennent quatre catégories d'axiomes:

-L'axiome de fermeture du domaine

Il indique que les seuls individus qui existent sont ceux que dénotent les constantes apparaissant dans les relations de la base (faits élémentaires et lois générales).

Exemple: pour la base
Presence(Anne)
Presence(Muriel)
Presence(Paul)
l'axiome de fermeture du domaine est:
(x) x=Anne v x=Muriel v x=Paul

Le rôle de l'axiome de fermeture du domaine est de permettre la validation de la formule (x) P(x) à partir de la satisfaction de P par tous les éléments du domaine. Cela correspond à l'induction.

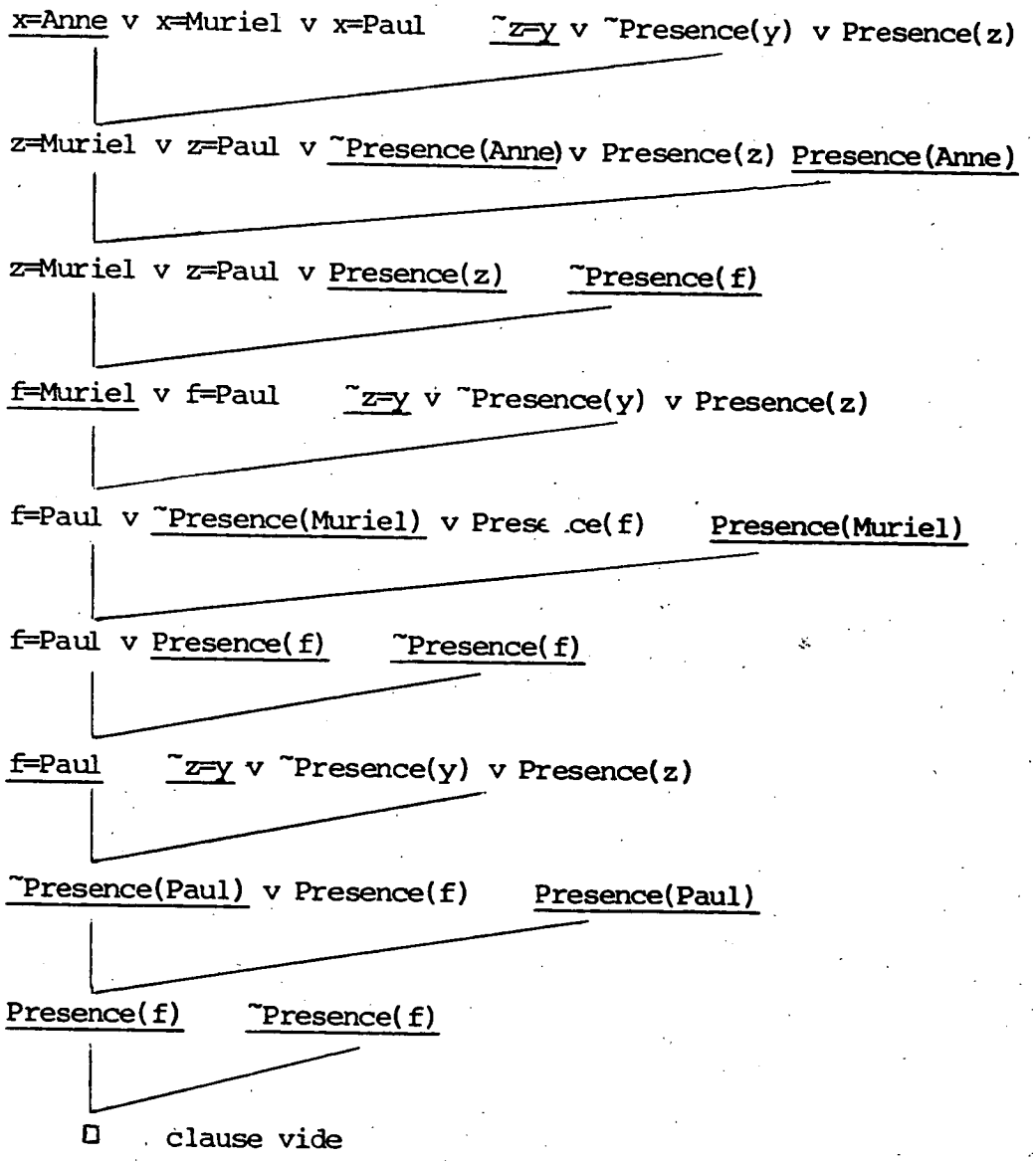
Exemple:

Dans l'exemple précédent, intuitivement, la question "Tous les invités sont-ils presents?" doit amener une réponse affirmative.
Or il est clair que la base ne permet pas de déduire la formule:
(x) Presence(x)

Si l'on ajoute à la base initiale l'axiome de fermeture du domaine
(x) x=Anne v x=Muriel v x=Paul
on obtient l'arbre de réfutation suivant

Question: (x) Presence(x)
Négation: Ex ~Presence(x) qui donne la clause:
~Presence(f)
(f fonction de skolem)

Un des axiomes de l'égalité (E4):
(y)(z) z=y & Presence(y) => Presence(z)
qui donne la clause
~z=y v ~Presence(y) v Presence(z)



-L'axiome d'unicité des noms

Cet axiome stipule que chaque entité syntaxique distincte dénote un individu distinct. En d'autres termes, il n'y a pas d'alias. Soit pour toute paire (a,b) de constantes distinctes a#b.

Dans l'exemple ci-dessus, l'axiome d'unicité des noms est
Anne#Muriel & Anne#Paul & Muriel#Paul

Sans cet axiome, il est impossible de prouver des formules telles que Anne#Paul, qu'il serait contre-intuitif de ne pas croire vraies.

-Les axiomes de complètion

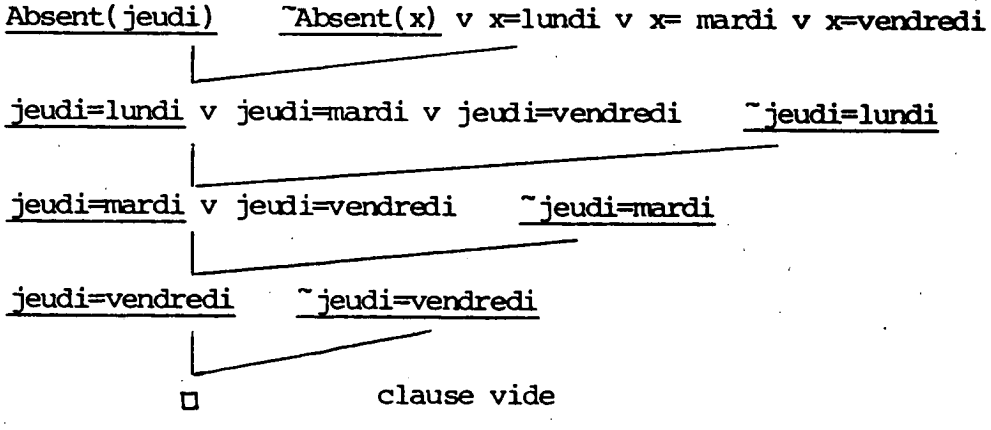
L'axiome de complètion d'un prédicat établit que les seuls individus qui satisfont ce prédicat sont ceux obtenus à partir des faits élémentaires et des règles de déduction.

Exemple: Si les seuls axiomes comprenant le prédicat Absent sont
Absent(lundi) v Absent(mardi)
Absent(vendredi)
L'axiome de complètion pour le prédicat Absent est
(x) Absent(x) => x=lundi v x=mardi v x=vendredi

Les axiomes de complètion sont destinés à prouver des formules négatives.

Exemple:
Ainsi, ~Absent(jeudi) sera démontré grâce à la contraposée de l'axiome de complètion:
(x) x#lundi & x#mardi & x#vendredi => ~Absent(x)
et aux clauses issues de l'axiome d'unicité des noms:
~jeudi=lundi
~jeudi=mardi
~jeudi=vendredi

Voici la réfutation détaillée pour la question ~Absent(jeudi)?



-Les axiomes de l'égalité

Ils sont classiquement regroupés selon les propriétés qu'ils expriment

-La réflexivité
E1: (x) x=x

-La symétrie
E2: (x) (y) x=y => y=x

-La transitivité
E3: (x) (y) (z) x=y & y=z => x=z

-La substitution de termes égaux
E4: Pour tout prédicat P, pour tous $x_1,\ldots,x_n,y_1,\ldots,y_n$
$P(x_1,\ldots,x_n)$ & $x_1=y_1$ &...& $x_n=y_n$ => $P(y_1,\ldots,y_n)$

L'introduction de ces axiomes est nécessité par le recours à l'égalité dont usent les axiomes de fermeture du domaine, d'unicité des noms et de complétion.

III 5 La base de données dans le système CIGARE

III 5 1 La base de données vue comme une théorie

La base de données est donc considérée comme définissant une théorie dont les axiomes sont les informations de la base (transcrites en formules du premier ordre) et les règles de particularisation.
Le traitement sémantique des données est amélioré en ce sens que les relations unaires sont identifiées à des classes sémantiques, comme dans Mc SKIMIN et MINKER [1977]. Ces types introduisent une information sémantique semblable à celle des sortes dans les logiques multi-sortes -une logique multi-sortes est une logique dans laquelle les variables sont quantifiées relativement à des domaines- (Pour une étude approfondie de l'influence des types dans le modèle relationnel, voir LACROIX et PIROTTE [1981]).

III 5 2 Le démonstrateur de théorèmes

Habituellement, la recherche des démonstrateurs de théorèmes est dirigée par des heuristiques, avec pour inconvénient la perte de la propriété de complétude (Ces démonstrateurs devant traiter des formules quelconques du premier ordre et apporter bien sûr une réponse en un temps fini, la seule solution est de sacrifier la complétude). Le saturateur étant décidable, il évite le recours aux heuristiques. En contrepartie, il n'opère que sur une classe réduite de formules, les clauses à variables prédéfinies. Les informations contenues dans la base de données devront donc respecter ce format. Cette exigence sera satisfaite grâce à l'existence des types et ceci sans dommage pour la généralité des informations enregistrables par la base.

La formule (qui doit être dépourvue de symboles fonctionnels) est d'abord traduite sous forme de clauses. Puis chacune de ces clauses $C(x_1,\ldots,x_n)$ est tranformée en une clause $\neg T_1(x_1)$ v...v $\neg T_n(x_n)$ v $C(x_1,\ldots,x_n)$ qui est une clause à variables prédéfinies. Chaque T_i est un prédicat qui impose le type de la variable x_i. En fait, la clause obtenue exprime la contrainte sur le type des variables:
$(x_1)\ldots(x_n)\ T_1(x_1)$ &...& $T_n(x_n)$ => $C(x_1,\ldots,x_n)$

IV LOGIQUE NON-MONOTONE

IV 1 Evolution d'une théorie axiomatique

Outre l'aspect interrogation, une base de données est aussi concernée par les opérations de mise à jour, soit ajout, retrait ou modification d'information. Puisqu'ici la base de données est une théorie axiomatique, le problème se ramène à l'évolution de cette théorie: que se passe-t-il lorsque des axiomes sont ajoutés? KOWALSKI [1979] identifie quatre cas.

(1) La nouvelle donnée est déjà un théorème

(2) La nouvelle donnée implique des données existantes

(3) La nouvelle donnée est indépendante des données existantes

(4) La nouvelle donnée est inconsistante avec la théorie courante

Les trois premières situations ne posent pas de problèmes majeurs car l'intégration de la nouvelle information respecte la propriété de monotonie de la logique du 1er ordre.

Monotonie de la logique des prédicats du 1er ordre:
Soient V et W deux ensembles de formules tels que V inclus dans W
alors pour toute formule w si V $\vdash$ w alors W $\vdash$ w

A l'inverse, la quatième situation est cause d'énormes difficultés. Les formules à l'origine de la contradiction doivent être identifiées afin de permettre la restauration de la consistance de la théorie. Cette tâche impliquant un comportement non-monotone de la théorie se révèle extrêmement délicate, au point que plusieurs alternatives ont été envisagées.

IV 2 Les situations

Pour contourner le problème d'inconsistance, HAYES [1971] a imaginé d'associer une théorie à chaque état de la base de données, chacun de ces états prenant le nom de "situation". Pour ce faire, est ajouté à chaque prédicat un argument qui indique dans quelle situation le prédicat est satisfait. Par exemple, Absent(Jean,s) signifie que dans la situation s Jean est absent.

Les transitions d'une situation à une autre sont formulées au moyen des axiomes du "cadre" (frame axioms), qui stipulent quelles formules demeurent vraies dans la nouvelle situation.

Exemple: Absent(Jean,s) => Absent(Jean, absence(Monique).s) est un axiome du cadre qui exprime que si Jean est absent dans une certaine situation s, alors il est également absent dans une situation identique où de plus Monique est absente.

Cette appréhension du problème du cadre (RAPHAEL [1971]) se heurte à de grosses difficultés d'implémentation. Elle nécessite qu'il y ait un axiome du cadre par prédicat et par transition, ce qui n'est pas toujours très réaliste. De plus la multiplicité des situations implique une croissance très importante soit de la taille de la base de données soit du temps de déduction. Une telle logique reste néanmoins monotone.

IV 3 Les TMS (Truth Maintenance System)

Un TMS (DOYLE [1979], Mc ALLESTER [1978]) gère un ensemble d'informations auxquelles une valeur de vérité est attachée. Un TMS possède quatre fonctions:

-il effectue une forme quelconque de déduction

-il maintient des justifications et explique le résultat de ses déductions

-il met à jour ses "croyances" lors d'ajout ou de retrait de prémisses

-il peut effectuer un retour arrière dirigé par la dépendance.

C'est cette dernière capacité qui est utile lors de la découverte d'une inconsistance, car elle permet de remonter jusqu'aux formules responsables de la contradiction. Ce point, allié au 2ème point confère un grand avantage aux TMS lors de l'ajout d'une information inconsistante avec la base, mais qui existait précédemment dans la base. Toutes les informations contradictoires avec cette "nouvelle" information sont en effet identifiées. Aucune déduction n'est donc nécessaire pour les retrouver.

IV 4 Une logique avec défauts

REITER [1980] a proposé une logique non-monotone dont le ressort principal concerne l'établissement de suppositions. Une supposition est une formule dont la validité est liée aux formules auxquelles elle est jointe: s'il est possible de déduire la formule $\neg w$ de ces formules W (i.e. $W \vdash \neg w$), alors w est considérée comme nulle et non avenue (schéma jusqu'à preuve du contraire supposer w). REITER [1980] appelle défauts les règles qui permettent de faire des suppositions.

IV 4 1 Les défauts

Un défaut est de la forme:

$$(x)\ \frac{A(x):\ MB1(x),\ldots,Bn(x)}{C(x)}$$

où A(x),B1(x),...,Bn(x) sont des formules du 1er ordre
M est l'opérateur modal de possibilité
si n=1 et si B(x) = C(x) le défaut est dit normal.

Seuls les défauts normaux possèdent une procédure de preuve, c'est pourquoi tous les défauts considérés ultérieurement appartiennent à cette classe.

Activation d'un défaut: Le défaut v: Mw donne à w la valeur d'un axiome si v est un théorème et si w est consistant avec la théorie.

Exemple:

$$(x)\ \frac{\text{lundi}(x) : M\ \text{Absence}(\text{Christian}, x)}{\text{Absence}(\text{Christian}, x)}$$

est un défaut qu'on peut interpréter comme: "si la date x tombe un lundi et s'il est consistant de supposer que Christian est absent à cette date x, alors inférer que Christian est absent à la date x".

Une théorie avec défauts normaux est un couple (D,W) où D est un ensemble de défauts normaux et W un ensemble de formules. Les extensions constituent le concept de base des théories avec défauts. Pour REITER, les défauts sont des règles qui étendent la théorie incomplète sous-jacente. Les défauts permettent d'ajouter à la théorie initiale un certain nombre de formules qu'il est possible de "croire" (pas forcément toutes en même temps, d'ailleurs). Les nouvelles théories ainsi obtenues sont des extensions de la théorie initiale.

Exemple:

$$D = \left\{ \frac{:M\ \text{Venir}(\text{Luc})}{\text{Venir}(\text{Luc})}\ ,\ \frac{:M\ \neg\text{Venir}(\text{Luc})}{\neg\text{Venir}(\text{Luc})} \right\} \quad W = \emptyset$$

Cette théorie a deux extensions
E1 = { Venir(Luc) } et E2 = {¬Venir(Luc) }.

$$D' = \left\{ \frac{\text{Venir}(\text{Yves})\ :M\ \neg\text{Venir}(\text{Anne})}{\neg\text{Venir}(\text{Anne})}\ ,\ \frac{:M\ \text{Venir}(\text{Yves})}{\neg\text{Venir}(\text{Yves})}\ ,\ \frac{\text{Venir}(\text{Noel})\ :M\ \text{Venir}(\text{Anne})}{\text{Venir}(\text{Anne})} \right\}$$

W' = { Venir(Noel), Venir(Anne) => Venir(Marc) }

Cette théorie avec défauts a aussi deux extensions:
E1' = { Venir(Noel), Venir(Anne) => Venir(Marc), Venir(Yves), ¬Venir(Anne) }
E2' = { Venir(Noel), Venir(Anne) => Venir(Marc), Venir(Yves), Venir(Anne) }
Dans cette extension le premier défaut n'a pas été activé car ¬Venir(Anne) n'est pas consistant. Ceci montre que l'ordre des défauts détermine les différentes extensions.

IV 4 2 La procédure de preuve

Contrairement à ce qui se passe pour la logique non-monotone de Mc DERMOTT et DOYLE [1980], une formule est un théorème d'une théorie avec défauts s'il existe une extension qui contienne cette formule. La procédure de preuve, démontrée saine et complète, est issue de la résolution. Les formules se présentent sous la forme de clauses indicées (C,{di}) où C est une clause et di un défaut.

Exemple: $d1 = \frac{:M\ Venir(Alice)}{Venir(Alice)} \rightarrow (Venir(Alice),\{d1\})$

$d2 = \frac{\neg Venir(Pierre)\ :M\ Venir(Jean)}{Venir(Jean)} \rightarrow (Venir(Jean),\{d2\})$

$(x)(y)\ Absence(x,y) \Rightarrow \neg Presence(x,y)$
$\rightarrow\ (\neg Absence(x,y) \vee \neg Presence(x,y),\ \emptyset)$

La résolvante de 2 clauses indicées (B,{b1,...,bn}) et (C,{d1,...,dm}) est la clause indicée
(R,{b1,...,bn} U {d1,...,dm}) où R est la résolvante habituelle de B et C.

Une formule w est prouvée par défaut relativement à une théorie avec défauts (D,W) si:

-il existe un réfutation de ($\neg$w,$\emptyset$) avec la réfutation sur les clauses indicées.

-il existe une réfutation des négations des prérequis des défauts employés dans la réfutation précédente (récursif).

-W et l'ensemble des conséquents des défauts employés dans les différentes réfutations sont consistants.

Exemple: soit la théorie avec défauts (D',W') décrite plus haut.
Une preuve par défaut de Venir(Marc) est :

($\neg$Venir(Marc),$\emptyset$) ($\neg$Venir(Marc) v Venir(Anne),$\emptyset$)

(Venir(Anne),$\emptyset$) (Venir(Anne),{d3})

($\square$,{d3})

Il faut maintenant trouver une réfutation du prérequis du 3ème défaut, soit Venir(Noel):

($\neg$Venir(Noel),$\emptyset$) (Venir(Noel),$\emptyset$)

($\square$,$\emptyset$)

Aucun défaut n'a été utilisé au cours de cette dernière réfutation, il ne reste plus qu'à vérifier la consistance de (W' U {d3}), qui est évidente.

IV 4 3 Evolution d'une théorie avec défauts

La propriété la plus intéressante se rapportant à l'évolution d'une théorie est la semi-monotonicité de cette logique.

Semi-monotonie: soient 2 théories T1=(D1,W1) et T2=(D1 U D2,W2). Alors toute extension de la théorie T1 est incluse dans une extension de la théorie T2.

C'est pourquoi la logique des défauts apporte une solution au problème de l'évolution d'une théorie. Bien entendu son application suppose que les formules de W soient des règles immuables et que l'information à assimiler soit représentée par un défaut.

Exemple:

$$D = \{ \frac{:M\ Absence(Luc,\ 28\ mars)}{Absence(Luc,\ 28\ mars)} \}$$

$$W = \{ (x)(y)\ Absence(x,y) \Rightarrow \neg Presence(x,y) \}$$

La théorie T=(D,W) possède une extension unique, dans laquelle ¬Presence(Luc, 28 mars) est vraie. Si l'information Presence(Luc, 28 mars), inconsistante avec la base de données, se présente, elle pourra être assimilée par la base. Il suffit que cette information soit codée en un défaut d et ajoutée à la base de données.

$$d = \frac{:M\ Presence(Luc,\ 28\ mars)}{Presence(Luc,\ 28\ mars)}$$

La nouvelle théorie T1=(D U {d}, W) possède 2 extensions. Une dans laquelle ¬Presence(Luc, 28 mars) est vraie (c'est la même extension que celle de T) et une où Presence(Luc, 28 mars) est vraie.

Le choix de l'extension dans laquelle il faut se placer résulte de la pertinence de l'information qui a été introduite dans la base de données. (Cette attitude s'accorde parfaitement avec la position de KOWALSKI [1979] qui avance que la restauration de consistance doit relever de comparaison sur l'"utilité" des informations incompatibles).

IV 4 4 Limites

Si, isolément, les défauts se comportent toujours correctement, des problèmes surgissent parfois, dûs à l'interaction de ces mêmes défauts (REITER, CRISCUOLO [1981]).

Les défauts sont transitifs, ce qui entraine quelquefois des dérivations inopportunes.

(x) $\frac{P(x) :M\ Q(x)}{Q(x)}$ et (x) $\frac{Q(x) :M\ R(x)}{R(x)}$ simulent (x) $\frac{P(x) :M\ R(x)}{R(x)}$

Pour bloquer cette transitivité, il faut rajouter le défaut

$$(x)\ \frac{P(x) :M\ \neg R(x)}{\neg R(x)}$$

opération qui correspond à faire une hypothèse (ou imposer une contrainte) sur les individus qui satisfont P.

Un autre ennui provient de défauts ayant des conséquents opposés.

Soient les défauts (x) $\frac{P(x) :M\ Q(x)}{Q(x)}$ et (x) $\frac{R(x) :M\ \neg Q(x)}{\neg Q(x)}$

Dans cette configuration, pour tout individu qui satisfait à la fois P et R, il existe une extension où il satisfait P, Q et R et une autre extension où il satisfait P, ¬Q et R. Pour se placer dans une extension où ni Q ni ¬Q ne sont satisfaits par un individu qui satisfait à la fois P et R il suffit d'ajouter les défauts:

(x) $\frac{P(x)\ \&\ \neg R(x) :M\ Q(x)}{Q(x)}$ et (x) $\frac{R(x)\ \&\ \neg P(x) :M\ \neg Q(x)}{\neg Q(x)}$

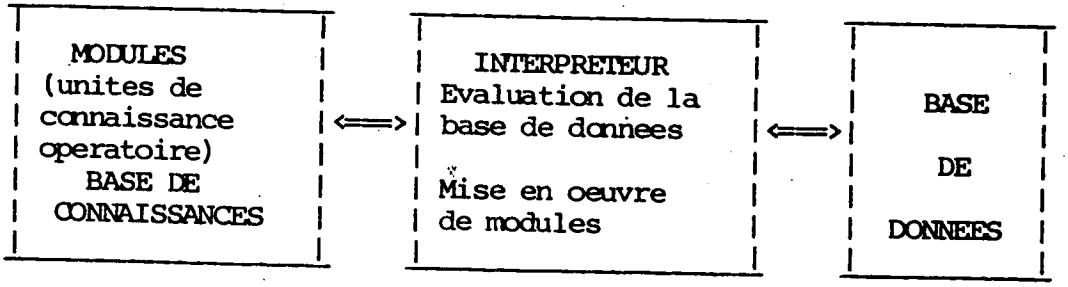
La difficulté est, bien entendu, de prévoir ces problèmes, puis de déterminer quels défauts supplémentaires doivent être introduits pour obtenir le comportement désiré.

V UN SYTEME EXPERT: CIGARE

V 1 Architecture du système

La finalité du système CIGARE est de prendre en charge tout ou partie des problèmes posés par l'organisation d'une réunion. Sa base de données recueille donc toutes les informations nécessaires pour atteindre cet objectif. Les informations sont très diversifiées, comparativement à leur nombre. En fait, la base de données se compose plutôt d'une collection de données hétéroclites. Ceci a pour conséquence qu'une appréhension correcte des différentes situations potentielles est extrêmement difficile. La raison en est le nombre très élevé d'information concourrant à la caractérisation d'un certain type de situation, chaque situation donnant lieu pratiquement à un traitement particulier.

Un système de production (WATERMAN, HAYES-ROTH [1978]) parait être la solution la plus raisonnable, mais aussi la plus adaptée en raison de la modularité de la connaissance opératoire. Un système de production effectue granulairement le traitement relatif à une situation donnée, en fonction des caractéristiques de la situation. Enfin cette approche simplifie le travail de conception en apportant une vue unificatrice entre l'analyse (des situations) et la synthèse (découpage des traitements assujettis aux résultats de l'analyse). Le système CIGARE obéit à la loi des trois composants des systèmes de production: base de données, base de connaissances, interpréteur.



V 2 La base de données

V 2 1 Les interlocuteurs

L'ensemble des interlocuteurs du système CIGARE régit, au moins partiellement, l'organisation de la réunion. C'est pourquoi les données associées à chacun d'eux et définissant l'attitude de l'individu vis-à-vis de la réunion sont particularisées.

Pour chaque réunion à laquelle il est invité, un interlocuteur se trouve caractérisé par l'ensemble de données suivant:

(1) Des réactions portant sur la valeur d'une réunion.

-contestation ou approbation du sujet, des dates...

-réponse positive à l'invitation

-...

(2) Un ensemble d'attributs propres à l'interlocuteur.

-son nom

-l'importance donnée à sa réponse

-...

Pour chaque interlocuteur il existe en plus une catégorie d'informations de niveau supérieur, les spécifications de comportement. Il ne s'agit plus cette fois de données destinées à l'organisateur de la réunion, mais au système lui-même. Par ce moyen l'utilisateur fournit au système des informations non seulement sur ses attitudes éventuelles, mais aussi sur ses actions éventuelles. "Si Pierre me demande si je vais à la réunion, répondez lui par la même question" est un exemple de spécification de comportement. De telles informations sont de même nature que les modules de la base de connaissances. C'est donc l'interpréteur lui-même qui se charge de leur exploitation.

V 2 2 Les autres données

Les informations restantes décrivent le but et les moyens de la réunion. Elles regroupent:

(1) Tout ce qui est "sémantique de la réunion".

-le sujet

-le lieu

-la liste des invités

-...

Ces données constituent le but de la réunion.

(2) Tout ce qui concerne exclusivement la préparation de la réunion.

-la date limite à laquelle les invités doivent avoir répondu

-les propositions soumises aux invités

-...

Ces données constituent les moyens de la réunion.

V 3 La base de connaissances

La base de connaissance est composée de modules ou unités de connaissance opératoires. Chaque module comprend une partie précondition et une partie action. Les actions sont des opérations à effectuer sur la base de données. Les préconditions sont des tests sur l'état de la base de données et s'expriment par des formules logiques car la base de données est elle-même constituée de formules. Une telle formule, soumise au système d'interrogation de la base de données, doit être satisfaite par la base de données pour que le module correspondant soit candidat à l'activation.

Les actions se résument à:

(1) La mise à jour de données.

-instanciation de la réponse d'un interlocuteur

-modification de cette réponse

-...

(2) L'interaction avec les utilisateurs.

-émission de courrier vers un ou plusieurs destinataires

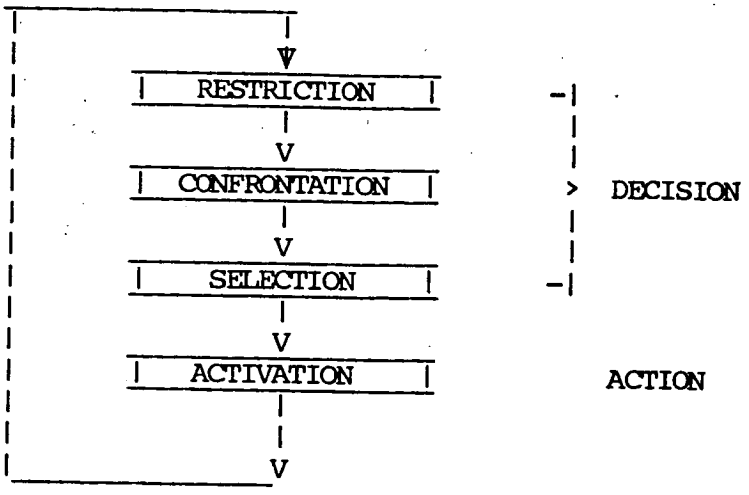
-...

L'ensemble des modules se divise en deux groupes, les modules qui s'appliquent à une réunion et les modules qui gèrent les interactions des différentes réunions entre elles.

V 4 L'interpréteur

L'interpréteur est le "moteur" d'un système de production. C'est lui qui est chargé de donner le contrôle au module approprié, puis une fois le travail de ce module achevé, de récupérer le contrôle. L'interpréteur fonctionne donc selon un cycle comportant une phase de décision (quel module activer) et une phase action (mise en oeuvre du module). Cette analyse et cette terminologie sont inspirées de FARRENY [1980].

Voici le détail du fonctionnement de l'interpréteur:



La phase de décision comporte trois étapes:

-la restriction permet de restreindre l'ensemble des modules candidats à la confrontation. Un système de production est d'autant plus performant que la restriction est sévère. En effet, la confrontation est très coûteuse du point de vue temps d'exécution, aussi moins il y a de modules à prendre en compte, plus la confrontation est efficace. Le principe de la restriction est de tirer parti de connaissances de niveau supérieur pour écarter le maximum de modules de la confrontation. Par exemple, lorsque l'interpréteur réagit à une demande d'information d'un utilisateur, il est inutile de vérifier si le module concernant l'annulation d'une réunion peut s'appliquer à la situation courante.

-la confrontation consiste à déterminer quels modules peuvent s'appliquer à la situation courante. Pour chaque module, l'interpréteur vérifie, en faisant appel au système d'interrogation de la base de données, si la formule représentant la précondition du module est satisfaite sur la base de données. Si oui, le module est dit activable (pour être effectivement activé il devra encore franchir l'épreuve de la sélection). Il s'agit donc de tester la validité d'une formule (la précondition) dans une théorie du premier ordre (la base de données). De par sa décidabilité, la saturation se présente comme la technique la plus adaptée pour résoudre ce problème.

-la sélection détermine quel module sera réellement activé parmi ceux dont la précondition s'est avérée satisfaite au cours de la confrontation. Ici aussi ce sont des connaissances de niveau supérieur qui conduisent à l'élimination des modules inopportuns. La hiérarchie des modules intervient notamment. Dans le système CIGARE, un module M1 est plus prioritaire qu'un module M2 si la situation à laquelle se rapporte le module M1 est un cas particulier de la situation à laquelle se rapporte M2 (i.e. les préconditions de M2 sont incluses dans celle de M1).

Exemple:

M1 Précondition: ~Venir(André)
Action: demander à André de réviser sa position

M2 Précondition: ~Venir(André) & ~Venir(Paul)
Action: annuler la réunion.

Il est bien clair que dans la situation où ni André ni Paul ne veulent venir, activer le module M1 causerait un dérangement inutile à André.

V 5 Un système d'aide à la décision

L'aptitude qu'ont les systèmes de production au raisonnement sur l'évolution du monde (raisonnement englobant plusieurs bases de données successives - la planification par exemple) se révèle profitable pour le système CIGARE. Ce dernier emploie principalement cette capacité pour une fonction d'aide à la décision.

Les techniques mises en jeu reposent sur une idée simple. Classiquement, les systèmes de production conservent l'état de plusieurs bases de données - plus exactement les états successifs d'une base de données - en enregistrant les différences entre chaque base. Seul l'état courant de la base de données est stocké explicitement. Quelques règles adéquates permettent d'exploiter pleinement cette représentation en restituant les différents états de la base, y compris l'état courant.

Dans CIGARE, les relations (appelées différences) entre chaque base de données sont représentées par leur position dans un arbre. Il est alors tout aussi possible de revenir à un état passé que de se placer dans une situation hypothétique.

Le système CIGARE peut ainsi, dans une certaine mesure, conseiller l'utilisateur amené à prendre un décision, en simulant les situations résultant des différents comportements possibles. Cette représentation est également source d'une fonction explicative. En cheminant parmi les états passés de la base de données, CIGARE peut permettre à l'utilisateur de mieux saisir les raisons de certains phénomènes (conséquence imprévue d'une de ses attitudes, par exemple).

VI INTERFACE

VI 1 Langue naturelle?

PETRICK [1976] rappelle les arguments en faveur et en defaveur de l'utilisation de la langue naturelle comme moyen d'interaction avec un système informatique. Le principal argument pour ce type d'utilisation est qu'un grand nombre des utilisateurs potentiels de l'informatique ne veut pas apprendre et par suite utiliser un langage formel ou artificiel. Par ailleurs PETRICK note que dans certaines applications la langue naturelle apparait comme le moyen idéal de communication et ceci quel que soit l'utilisateur.

Le sytème CIGARE s'inscrit dans l'optique de l'automatisation des tâches de bureau et s'adresse donc à des gens ayant des expériences et compétences très diverses en informatique. Pour mettre chacun sur un pied d'égalité devant l'utilisation du système et en nous conférant aux arguments ci-dessus, il nous a semblé utile de doter CIGARE d'une interface en langue naturelle.

Toutefois, ainsi que le note HAYES et al. [1981], ce n'est pas tant l'utilisation de la langue naturelle qui est importante pour obtenir une interface amicale que les caractéristiques du dialogue homme-machine. Ainsi l'utilisation de plusieurs moyens de communication (écrans préformattés, multifenêtres, sensibles, souris, parole...) peut donner une souplesse d'utilisation aussi grande que celle de la langue naturelle. En faveur de cette dernière on citera encore WALTZ [1978] qui souligne la plus grande richesse de la langue naturelle et les facilités qu'elle apporte pour vérifier l'exactitude d'une compréhension. Ainsi certaines requêtes à une base de données ne peuvent être obtenues à partir d'un menu, par exemple la requête:
"est-ce que des avions ayant eu une révision de moteur en mai ont volé 10 heures ou moins en juin?".

Dans CIGARE on essaie de définir un langage d'accès qui puisse se dégrader du naturel vers un langage de commande (en forme d'expressions logiques, par exemple). Un dialogue dirigé assure actuellement l'interface et des essais d'intégration de matériels de reconnaissance et de synthèse de parole sont en cours.

VI 2 Analyse de la langue naturelle par la logique

Nous avons vu que modèle adopté pour la représentation de la base de données des faits dans CIGARE était le modèle logique. De même pour représenter la sémantique des phrases d'entrée nous avons choisi la logique des prédicats du 1er ordre. Elles sont

ainsi directement utilisables par l'interpréteur de commandes qui n'est autre qu'un démonstrateur de théorèmes.

KOWALSKI [1979] a montré que la logique (le sous-ensemble des clauses de HORN) peut servir de langage de programmation. Ce langage possède une sémantique bien définie ce qui n'est pas le cas, par exemple, des réseaux sémantiques pour lesquels DELIYANNI et KOWALSKI [1979] ont montré qu'une extension pouvait se représenter sous forme de clauses de HORN. La structure du réseau n'est en fait utile que pour optimiser la résolution.

Par ailleurs, les grammaires d'analyse de langue naturelle peuvent être représentées élégamment et efficacement dans le formalisme de la logique. Les règles sont des formules logiques et l'inférence logique est utilisée pour relier la forme textuelle à la représentation sémantique. Plusieurs formalismes basés sur PROLOG ont été proposés:

-les grammaires de métamorphose par COLMERAUER [1978].

-les grammaires à clauses definies (clauses ne comportant qu'un seul littéral positif) formant un sous-ensemble des précédentes par PEREIRA et WARREN [1980]. Ces derniers ont également montré qu'elles sont aussi puissantes que les réseaux de transition augmentés (ATN) de WOODS [1970] et que ces mêmes ATN sont facilement traduisibles en une telle grammaire. A l'avantage des grammaires à clauses définies la clarté, la flexibilité et les bases pour un travail théorique.

-les grammaires d'extraposition par PEREIRA [1980] intègrent les grammaires à clauses définies et permettent, en particulier, de représenter de manière élégante le déplacement de l'antécédent dans une relative (extraposition). Ce formalisme permet, en outre de résoudre simplement les contraintes syntaxiques sur les groupes nominaux (par exemple la contrainte complexe sur les groupes nominaux de ROSS).

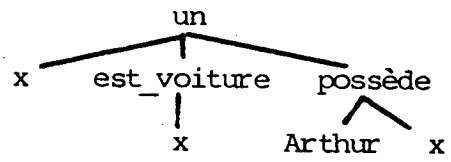
-en dehors de PROLOG on notera l'effort de CHARNIAK [1981] et WONG [1981] d'aboutir à un formalisme commun à la resolution de problèmes et à la compréhension du langage naturel. Ce formalisme inclut le calcul des prédicats et les frames. Les facettes (slots) des frames sont représentées par des formules logiques. L'avantage est de partitionner l'espace des connaissances.

Au niveau des applications on peut citer différents travaux:

-COLMERAUER [1979], DAHL [1977], KITTREDGE [1980] présentent des systèmes d'interrogation de bases de données en langage naturel. Les phrases sont traduites en formules logiques avec une attention spéciale dévolue aux articles. Ceux-ci sont représentés par des quantificateurs a trois branche indiquant le fait que ce quantificateur lie une variable à deux formules:

"Arthur possède une voiture"

est représenté par:



Les formules sont ensuite évaluées dans le cadre d'une logique à trois états (vrai, faux, indéfini). En outre les formules sont telles que les propriétés ne portent pas sur des individus isolés mais sur des ensembles d'individus.

-sur un sous-ensemble restreint de langue naturelle SABATIER [1980] s'attache à dégager les présuppositions et les assertions induites par une phrase. Une phrase n'est déclarée correcte que si les présuppositions induites sont consistantes aves les faits stockés dans la base de connaissance. Les assertions peuvent mener à la modification des faits déjà stockés. Cette modification consiste en un calcul de situation (HAYES [1971]).

-Mc CORD [1981,1982] construit lors de l'analyse par une grammaire à clauses définies un arbre original. Chaque noeud de l'arbre représente une entité syntaxique. Cette entité syntaxique se présente sous la forme d'une structure à quatre champs: les informations syntaxiques et morphologiques, un déterminant montrant comment l'entité peut modifier une autre entité syntaxique, un prédicat central (sens du verbe ou du nom dans un VP ou NP) et enfin une liste de modificateurs qui sont eux-mêmes des structures à quatre champs. Les règles de syntaxe utilisent la notion de facettes (slot) pour controler la complémentation des verbes, noms et adjectifs. Les facettes peuvent avoir des types sémantiques (précisés dans le lexique) qui rendent l'analyse très efficace (DAHL [1979]). Des règles d'interprétation sémantique traduisent la structure arborescente en une formule logique en déterminant la portée des modificateurs et des quantificateurs.

SILVA et DWIGGINS [1981] proposent un analyseur de textes. Chaque phrase ou texte est décomposé en évènements atomiques. Ceux-ci vont remplir une structure de données évènement qui regroupe l'ensemble des états, processus et actions associés avec un objet. Les évènements atomiques sont des instances de frames représentées par des procédures PROLOG (paquet de clauses).

VI 3 Solutions retenues pour CIGARE

L'analyseur syntaxico-sémantique de l'interface homme-machine du système CIGARE est très largement inspiré de COLMERAUER [1979]. Dans son modèle (cf. plus haut) aux noms communs, adjectifs et verbes sont associés des propriétés a n arguments et à chaque article un quantificateur à trois branches q(x,f1,f2) signifiant pour q x tel que f1 il est vrai que f2. COLMERAUER introduit également la négation ainsi que les relatives restrictives. Pour CIGARE il s'est avéré nécessaire

d'étendre le formalisme aux compléments circonstanciels de temps et de lieu. par ailleurs nous souhaitons que CIGARE puisse donner la raison de ses actions. Il doit pouvoir, pour ce faire, analyser les questions du type pourquoi, où, quand, comment. Pour chaque nom et verbe est stockée, dans le lexique, la liste des compléments ou sujet possibles ainsi que leur type sémantique. Ceci permet de faire une vérification sémantique précoce lors de l'analyse syntaxique.

Au niveau de la gestion du dialogue, résolution des ellipses et anaphores nous pensons utiliser CADI (NOUHEN-BELLEC, SIROUX [1981]). La liaison est en effet très simple car CADI reçoit en entrée une suite de n-uplets qui peut être considérée comme une conjonction de formules atomiques. Toutefois la limitation viendra sans doute du fait qu'on ne peut utiliser que ce type de formules.

Il nous semble également intéressant d'introduire au niveau de la compréhension un module de séquencement des actions inspiré de SHANK et ABELSON [1977]. Suivant les plans entrepris précédemment un certain utilisateur ne peut exécuter qu'un nombre d'actions limité. Ceci permet de diminuer le nombre d'inférences et d'exercer un contrôle sur les actions entreprises par l'utilisateur. Cela permet également de venir en aide à l'utilisateur en lui indiquant quelles sont les actions possibles à partir de l'étape où il est rendu.

CONCLUSION

Nous avons montré comment le formalisme de la logique des prédicats du 1er ordre pouvait être utilisé pour représenter un système expert. Nous avons par ailleurs montré qu'il était possible d'utiliser la non-monotonie de la logique des défauts pour représenter l'évolution d'une base de données.

Au niveau réalisation, un démonstrateur de théorèmes raisonnant également par défauts a été implémenté en PROLOG. L'analyseur syntaxico-sémantique du langage naturel ainsi que CADI est en cours d'intégration à une maquette du système CIGARE (maquette ancienne, tout-à-fait différente des idées exprimées ici, programmée en PASCAL et ayant servi à l'évaluation du service fourni par l'application).

Actuellement, nous portons notre effort sur la production de règles expertes modélisant la préparation de réunions. Nous essayons également d'étendre la classe des formules acceptables par la base de données. Rappelons qu'actuellement, seules les formules à variables prédéfinies sont acceptées et que celles-ci ne peuvent pas comporter de fonctions.

BIBLIOGRAPHIE

BOSSU G., SIEGEL P. (mai 1981) La saturation au secours de la non-monotonie.
Thèse de 3ème cycle. Université d'Aix-Marseille II.

BRODIE M.L., ZILLES S.N. (eds) (juin 1981) Proc. workshop on data abstraction, databases and conceptual modelling.
ACM SIGART, SIGMOD, SIGPLAN. Pingree Park, Colorado.

CHANG C.L., LEE R.C.T (1973) Symbolic logic and mechanical theorem-proving.
Academic Press.

CHARNIAK E. (juil. 1981) A common representation for problem-solving and language comprehension information.
Artificial Intelligence 16,3.

CHURCH A. (1936) A note on the Entschreidungsproblem.
Journ. Symbolic logic, vol. 1 p 40-41. Correction idem p 101-102.
Reimprimé dans DAVIS 1965.

CLARK K.L. (1978) Negation as failure.
dans GALLAIRE, MINKER (eds) Logic and databases, Plenum Press.

CODD E.F. (juin 1970) A relational model of large shared data banks.
CACM 13,6.

CODD E.F. (dec. 1979) Extending the relational model to capture more meaning.
ACM Transaction on database systems 4,4.

COLMERAUER A. (1978) Metamorphosis grammars.
dans Natural language communication with computers, L. BOLC (ed) Lecture notes in computer science, Springer Verlag.

DAHL V. (nov.1977) Un système dèductif d'interrogation de base de données en espagnol.
Thèse de 3ème cycle. Université d'Aix-Marseille II.

DAHL V. (aout 1979) Quantification in a three-valued logic for natural language question-answering systems.
6th IJCAI, Tokyo.

DAVIS M. (ed) (1965) The undecidable basic papers on undecidable propositions, unsolvable problems and computable fonctions.
Hewlitt, Raven Press, N.Y.

DELIYANNI A., KOWALSKI R.A. (mars 1979) Logic and semantic networks.
CACM 22,3.

DOYLE J. (nov. 1979) A truth maintenance system.
Artificial Intelligence 12,3.

FARRENY H. (sept. 1980) Un système pour l'expression et la résolution de problèmes orienté vers le contrôle de robots
Thèse d'état. Université Paul Sabatier. Toulouse.

GALLAIRE H., MINKER J. (eds) (1978) Logic and databases.
Plenum Press, N.Y.

GODEL K. (1930) Die Vollstandigkeit der Axiome des logischen Funktionenkalkuls.
Monathsh. Math. Phys. vol 37 p 349-360. Traduction anglaise The completeness of the axioms of the functional calculus of logic, dans From FREGE to GODEL, J. Van HEIGENVORT (ed), Harvard University Press, Cambridge, Mass. 1967.

HAYES P.J. (1971) A logic of actions.
dans Machine Intelligence 6, MELTZER, MITCHIE (eds).

HAYES P., BALL E., REDDY R. (mars 1981) Breaking the man-machine communication barrier.
Computer.

HILBERT D., ACKERMANN W. (1928) Grundzunge des theoretischen Logik.
Springer Verlag. Traduction anglaise Principles of mathematical logic, Chelsea Pub co., N.Y. (1950).

KITTREDGE R. (mai 1980) Natural language queries for a linguistic database using PROLOG.
Proc. CSCSI/SCEIO conference, Victoria.

KOWALSKI R.A. (1979) Logic for problem-solving.
North-Holland.

KOWALSKI R.A., HAYES P.J. (1969) Semantic trees in theorem-proving.
Machine Intelligence 4 p 87-101, MELTZER, MITCHIE (eds).

LACROIX M., PIROTTE A. (1980) Associating types with domains of relational databases.

dans BRODIE et ZILLES.

LIPSKI W.Jr. (janv. 1981) On database with incomplete information.
JACM 28,1.

LOWENHEIM L. (1915) Uber Mogligkeiten im Relationkalkul.
Math. ann. vol. 76, p 447-470. Trad. anglaise dans Van HEIGENVORT (1967).

LYNDON R.C. (1964) Notes on logic.
Von NOSTRAND Mathematical Studies.

Mc ALLESTER D.A. (1978) A three-valued truth maintenance system.
MIT Lab. memo 473.

Mc CORD M.C. (mai 1982) Using slots and modifiers in logic grammars for natural language.
Artificial Intelligence 18,3.

Mc CORD M.C. (1981) Focalizers, the scoping problem and semantic interpretation rules in logic grammars. Technical report 81-81. University of Kentucky.

Mc DERMOTT D., DOYLE J. (avril 1980) Non-monotonic logic I.
Artificial Intelligence 13,2.

Mc SKIMIN J.R., MINKER J. (1977) The use of a semantic network in a deductive question-answering system.
5th IJCAI, Cambridge Mass.

MENDELSON E. (1964) Introduction to mathematical logic.
Von NOSTRAND, N.Y.

NOUHEN-BELLEC A., SIROUX J. (fev. 1981) CADI: constructeur automatique de dialogues intelligent intègrè à un système de reconnaissance de la parole.
Thèse de 3ème cycle. Université de Rennes.

PEREIRA F. (juil. 1980) Extraposition grammars.
Logic programming workshop, Debrecen Hongrie.

PEREIRA F., WARREN D. (mai 1980) Definite clause grammars for language analysis - A survey of the formalism and a comparison with augmented transition networks.
Artificial Intelligence 13,3.

PEREIRA F., PEREIRA L., WARREN D. (1978) User'guide to DEC-System 10 PROLOG.
Dept. of A.I., University of Edinburgh.

PETRICK S.R. (1976) On natural language based computer systems.
IBM J. of research and development.

RAPHAEL B. (1971) The frame-problem in problem-solving systems.
dans Artificial Intelligence and Heuristic Programming, FINDLER, MELTZER (eds) Edinburgh University Press.

REITER R. (1978) On closed world databases.
dans GALLAIRE, MINKER Logic and databases.

REITER R. (avril 1980) A logic for default reasoning.
Artificial Intelligence 13,2.

REITER R., CRISCUOLO G. (aout 1981) On interacting defaults.
7th IJCAI, Vancouver.

ROBINSON J.A. (1965) A machine-oriented logic based on the resolution principle.
JACM 12, p 23-41.

SABATIER P. (juin 1980) Dialogues en français avec un ordinateur.
Thèse de 3ème cycle. Université d'Aix-Marseille II.

SHANK R.C., ABELSON R.P. (1977) Scipts, plans, goals and understanding.
LEA Press. N.Y.

SILVA G., DWIGGINS D. (oct. 1980) Towards a prolog text grammar.
Sigart Newsletter.

TURING A.M. (1936) On computable numbers with an application to the Entscheidungsproblem.
Proc. London Math. soc. 42(2) 1936-7 p 230-265. A correction idem 43 1937 p 544-546. Reimprimé dans DAVIS 1965 p 115-154.

WALTZ D.L. (juil. 1978) An english question-answering system for a large relational database.
CACM 21,7.

WATERMAN D.A., HAYES-ROTH F (eds) (1978) Pattern-directed inference systems.
Academic Press.

WONG D. (aout 1981) On the unification of language comprehension with problem-solving.
Ph D. thesis. Dept. of computer science. Brown University.

WOODS W.A. (oct. 1970) Transition network grammars for natural language analysis.
CACM 13.

YAZDANIAN K. (1976) Problèmes de décision dans les théories du 1er ordre.
Rapport interne LBD 76-8 ONERA-CERT Toulouse.

Liste des Publications Internes IRISA

PI 155 **Analyses d'opinions d'instituteurs à l'égard de l'appropriation des nombres naturels par les élèves de cycle préparatoire**
R. Gras , 37 pages ; *Octobre 1981*

PI 156 **Récursion induction principle revisited**
G. Boudol, L. Kott , 49 pages ; *Décembre 1981*

PI 157 **Loi d'une variable aléatoire à valeur R^*** réalisant le minimum des moments d'ordre supérieur à deux lorsque les deux premiers sont fixés
M.Kowalowka, R. Marie , 8 pages ; *Décembre 1981*

PI 158 **Réalisations stochastiques de signaux non stationnaires, et identification sur un seul échantillon**
A. Benveniste J.J. Fuchs , 33 pages ; *Mars 1982*

PI 159 **Méthode d'interprétation d'une classification hiérarchique d'attributs-modalites pour l'«explication» d'une variable ; application à la recherche de seuil critique de la tension artérielle systolique et des indicateurs de risque cardiovasculaire**
B. Tallur , 34 pages ; *Janvier 1982*

PI 160 **Probabilite stationnaire d'un réseau de files d'attente multiclasse à serveur central et à routages dépendant de l'état**
L.M. Le Ny , 18 pages ; *Janvier 1982*

PI 161 **Détection séqentielle de changements brusques des caractéristiques spectrales d'un signal numérique**
M. Basseville, A. Benveniste , pages ; *Mars 1982*

PI 162 **Actes regroupés des journées de Classification de Toulouse (Mai 1980), et de Nancy (Juin 1981)**
I.C. Lerman , 304 pages ;

PI 163 **Modélisation et Identification des caractéristiques d'une structure vibratoire : un problème de réalisation stochastique d'un grand système non stationnaire**
M. Prévosto, A. Benveniste, B. Barnouin , 46 pages ; *Mars 182*

PI 164 **An enlarged definition and complete axiomatization of observational congruence of finite processes**
Ph. Darondeau , 45 pages ; *Avril 1982*

PI 165 **Accès vidéotex à une banque de données médicales**
A. Chauffaut, M. Dragone, R. Rivoire, J.M. Roger , 25 pages ; *Mai 1982*

PI 166 **Comparaison de groupes de variables définies sur le même ensemble d'individus**
B. Escofier, J. Pages , 115 pages ; *Mai 1982*

PI 167 **Transport en circuits virtuels internes sur réseau local et connexion Transpac**
M. Tournois, R. Trépos , 90 pages ; *Mai 1982*

PI 168 **Impact de l'intégration sur le traitement automatique de la parole**
P. Quinton , 14 pages ; *Mai 1982*

PI 169 **A systolic algorithm for connected word recognition**
J.P. Banâtre, P. Frison, P. Quinton , 13 pages ; *Mai 1982*

PI 170 **A network for the detection of words in continuous speech**
J.P. Banâtre, P. Frison, P. Quinton , 24 pages ; *Mai 1982*

PI 171 **Le langage ADA : Etude bibliographique**
J. André, Y. Jégou, M. Raynal , 12 pages ; *Juin 1982*

PI 172 **Comparaison de groupes de variables : 2ème partie : un exemple d'application**
B. Escofier, J. Pajès , 37 pages ; *Juillet 1982*

PI 173 **Unfold-fold program transformations**
L. Kott , 29pages ; *Juillet 1982*

PI 174 **Remarques sur les langages de parenthèses**
J.M. Autebert, J. Beauquier, L. Boasson, G. Senizergues , 20 pages ; *Juillet 1982*

PI 175 **Langages de parenthèses, langages N.T.S. et homomorphismes inverses**
J.M. Autebert, L. Boasson, G. Senizergues , 26 pages ; *Juillet 1982*

PI 176 **Tris pour machines synchrones ou Baudet Stevenson revisited**
R. Rannou , 26 pages ; *Juillet 1982*

PI 177 **Un nouvel algorithme de classification hiérarchique des éléments constitutifs de tableau de contingence basé sur la corrélation**
B. Tallur , Juillet 1982 ;

PI 178 **Programmes d'analyse des re'sultats d'une classification automatique**
I.C. Lerman et collaborateurs , 79 pages ; *Septembre 1982*

PI 179 **Attitude à l'égard des mathématiques des élèves de sixième**
J.Degouys, R. Gras, M. Postic , 29 pages ; *Septembre 1982*

PI 180 **Traitements de textes et manipulations de documents : bibliographie analytique**
J. André , 20 pages ; *Septembre 1982*

PI 181 **Algorithme assurant l'insertion dynamique d'un processeur autour d'un réseau à diffusion et garantissant la cohérence d'un système de numérotation des paquets global et réparti**
Annick Le Coz, Hervé Le Goff, Michel Ollivier , 31 pages ; *Octobre 1982*

PI 182 **Interprétation non linéaire d'un coefficient d'association entre modalités d'une juxtaposition de tables de contingence**
Israël César Lerman , 34 pages ; *Novembre 1982*

PI 183 **L'IRISA vu à travers les stages effectués par ses étudiants de DEA (1è année de thèse)**
Daniel Herman , 41 pages ; *Novembre 1982*

PI 184 **Commande non linéaire robuste des robots manipulateurs**
Claude Samson , 52 pages ; *Janvier 1983*

PI 185 **Dialogue et représentation des informations dans un système de messagerie intelligent**
Philippe Besnard, René Quiniou, Patrice Quinton, Patrick Saint-Dizier, Jacques Sirou, Laurent Trilling , 45 pages ; *Janvier 1983*

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique